

Lenovo Educational Board Intermediate Manual

Lenovo™

Contents

Components Schematic.....	5
Lesson I – 1: Turn On The Nightlight	6
OVERVIEW.....	6
Teacher Guide.....	7
Background Info.....	8
Procedure.....	8
Result	11
Lesson I – 2: Clap On, Clap Off	12
OVERVIEW.....	12
Teacher Guide.....	13
Background Info.....	14
Procedure.....	14
Result	16
Lesson I – 3: Control LEDs With Wifi.....	17
OVERVIEW.....	17
Teacher Guide.....	18
Background Info.....	19
Procedure.....	20
Result	24
Lesson I – 4: Can You Feel the Heat?.....	26
OVERVIEW.....	26
Teacher Guide.....	27
Background Info.....	28
Procedure.....	28
Result	31
Lesson I – 5: Morse Code.....	32
OVERVIEW.....	32
Teacher Guide.....	33
Background Info.....	34
Procedure.....	36
Result	43
Lesson I – 6: Proximity Sensor	44
OVERVIEW.....	44

Teacher Guide.....	45
Background Info.....	46
Procedure.....	46
Result	49
Lesson I – 7: Mini Sound Board.....	50
OVERVIEW.....	50
Teacher Guide.....	51
Background Info.....	52
Procedure.....	52
Result	55
Lesson I – 8: Motion Detector	56
OVERVIEW.....	56
Teacher Guide.....	57
Background Info.....	58
Procedure.....	58
Result	60
Lesson I – 9: Guess the Number	62
OVERVIEW.....	62
Teacher Guide.....	63
Background Info.....	64
Procedure.....	64
Result	67
Now we have a fully functioning random number guessing game where you have 5 attempts to guess a number that the computer is thinking of.	67
Lesson I – 10: Create Your Own Timer	68
OVERVIEW.....	68
Teacher Guide.....	69
Background Info.....	70
Procedure.....	70
Result	74
Lesson I – 11: Bluetooth LED Control	75
OVERVIEW.....	75
Teacher Guide.....	76
Background Info.....	77
Procedure.....	77
Result	84

Lesson I – 12: OLED Display Temperature 85

OVERVIEW.....85

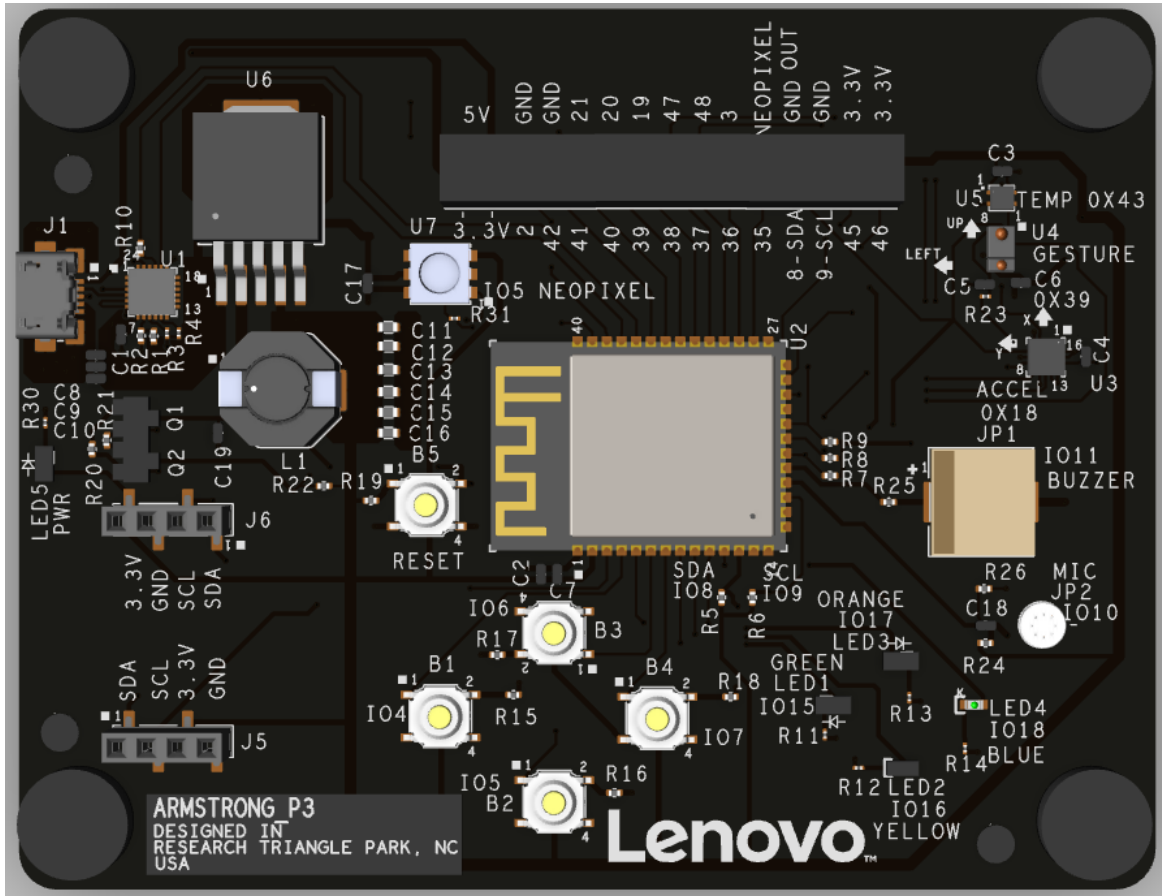
Teacher Guide.....86

Background Info.....87

Procedure.....87

Result88

Components Schematic



Reference ID	Component	Function
B1	Button 1	Input button
B2	Button 2	Input button
B3	Button 3	Input button
B4	Button 4	Input button
B5	Button 5	Reset
IO15	LED 1	Green LED Output
IO16	LED 2	Yellow LED Output
IO17	LED 3	Orange LED Output
IO18	LED 4	Blue LED Output
IO10	Microphone	Audio Input/Output
IO11	Buzzer	Audio Output
IO1	Neopixel	LED Output
J1	Port 1	USB port
L1		Inductor
U1		
U2	CPU	
U3	Accelerometer	
U4	Gesture Sensor	
U5	Temperature Sensor	
U6		
U7	Neopixel	

Lesson I – 1: Turn On The Nightlight



OVERVIEW

In this lesson we will:

- Combine the use of a light sensor and an LED to create a nightlight

Teacher Guide

In this lesson, the students will be combining their skills that they learned about the Neopixel and the ambient light sensor to create a night light. Most of the content in this lesson has been previously learned, but we are combining these sections of code together to create a new use.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

Wouldn't it be convenient if your night light could automatically turn on when you turn out the light or off when you turn on the light? With a light sensor, we can utilize its functionality to automate the switching of the LED light when there is little to no light present in the room.



Many modern night lights are designed to turn on automatically when it gets dark and turn off automatically when it is light again. They use a detector called a CdS cell. A CdS cell is a type of resistor called a **photoresistor**. Resistors reduce the flow of electricity. Photoresistors differ from other resistors in that they change when light shines on them. When light shines on a CdS cell, it decreases the resistance. The night light has a detector circuit which automatically turns on the light when the resistance reaches a certain level. High resistance means that there is no light shining on the CdS cell, so the night light turns on.

Procedure

Since the Lenovo Educational board is equipped with a light sensor and a Neopixel LED, we will take advantage of these components to create our own night light!

1. Create a blank sketch and save it as "Turn on the Nightlight."
2. Include the following libraries and variables as well as create the following objects:

```
// include the following libraries
#include <Wire.h>
#include <SparkFun_APDS9960.h>
#include <Adafruit_NeoPixel.h>

#define NEOPIXEL_PIN 1 // Neopixel is on pin 1
#define NEOPIXEL_COUNT 1 // There is only 1 RGB LED on DOS
#define NEOPIXEL_INDEX 0 // RGB LED is at index 0
#define DELAYVAL 500 // Time (in milliseconds) to pause between pixels
uint16_t ambient_light = 0;

// Create objects
Adafruit_NeoPixel Pixel(NEOPIXEL_COUNT, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);
SparkFun_APDS9960 apds = SparkFun_APDS9960();
```


3. Input the following `setup()` function to initialize Neopixel and APDS:

```
void setup() {
  // Initialize Serial port
  Serial.begin(9600);
  Serial.println();

  // Initialize APDS
  apds.init();
  apds.enableLightSensor(false);
  // Wait for initialization and calibration to finish
  delay(500);

  // Initialize Neopixel
  Pixel.begin();
  // Set all pixel colors to OFF
  Pixel.clear();
  Pixel.show();
}
```

4. Input the following `loop()` function find ambient light values and set the appropriate brightness for the Neopixel:

```
void loop() {
  // Read the light levels (ambient)
  apds.readAmbientLight(ambient_light);
  // if statements to turn Neopixel brightness
  // based on ambient light values
  if (ambient_light < 15)
  {
    Pixel.setPixelColor(NEOPIXEL_INDEX, 255, 255, 255);
    Pixel.show();
  }
  else if (ambient_light >= 15 && ambient_light < 30)
  {
    Pixel.setPixelColor(NEOPIXEL_INDEX, 250, 250, 250);
    Pixel.show();
  }
  else if (ambient_light >= 30 && ambient_light < 50)
  {
    Pixel.setPixelColor(NEOPIXEL_INDEX, 245, 245, 245);
    Pixel.show();
  }
  else if (ambient_light >= 50 && ambient_light < 75)
  {
    Pixel.setPixelColor(NEOPIXEL_INDEX, 235, 235, 235);
    Pixel.show();
  }
  else if (ambient_light >= 75 && ambient_light < 100)
  {
    Pixel.setPixelColor(NEOPIXEL_INDEX, 220, 220, 220);
    Pixel.show();
  }
}
```

```
else if (ambient_light >= 100 && ambient_light < 200)
{
    Pixel.setPixelColor(NEOPIXEL_INDEX, 190, 190, 190);
    Pixel.show();
}
else if (ambient_light >= 200 && ambient_light < 1000)
{
    Pixel.setPixelColor(NEOPIXEL_INDEX, 100, 100, 100);
    Pixel.show();
}
else if (ambient_light >= 1000 && ambient_light < 3000)
{
    Pixel.setPixelColor(NEOPIXEL_INDEX, 50, 50, 50);
    Pixel.show();
}
else if (ambient_light >= 3000 && ambient_light < 5000)
{
    Pixel.setPixelColor(NEOPIXEL_INDEX, 25, 25, 25);
    Pixel.show();
}
else
{
    Pixel.setPixelColor(NEOPIXEL_INDEX, 0, 0, 0);
    Pixel.show();
}
}
```

Result

You have just created an automatic nightlight! Try testing it by covering the gesture sensor with your hand.

Now It's Your Turn!

- Can you program the nightlight so you can turn it on/off with a button?
- Can you program the nightlight so that it lights up different LEDs according to brightness?

Lesson 1 – 2: Clap On, Clap Off



OVERVIEW

In this lesson we will:

- Write a simple program to use clapping noise to turn on and off a LED

Teacher Guide

In this lesson, students will interact with the microphone and a LED to create a device that resembles a smart lightbulb. One of the difficult things for this lesson will be getting the claps to register. For us, a micVal of 30 gave the best results, but this value can be adjusted lower (i.e., 20 or 25 to register quieter claps). Additionally, between claps when turning off the LED make sure that there is a half second pause. Meanwhile, after turning off the LED, make sure students wait at least a second before trying to turn it back on again. Finally, if you use sounds other than claps, it may not register turning on and off the LED correctly.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

Many modern homes are equipped with automatic gadgets to provide a better living experience for those who live there. You have probably heard of an automatic light that turns on/off when you make a clapping noise. One clap would turn on the light and two claps would turn off the lights.

This works because the smart light is equipped with a sound sensor. The sound sensor is programmed to switch on/off the light when a clapping noise is made. The sound detector will act as a switch to control the light. In this lesson, we will teach you how to use the noise detector with the Lenovo Educational board to create your own automatic clapping light.



Procedure

1. Create a new sketch and input the following variables and constants above the setup() function:

```
const int BlueledPin = 18; // blue LED pin
const int micPin = 10; // microphone pin number
int micVal = 0; // value read from the microphone
bool litUp = false; // tells whether the LED is on/off
```

2. Input the following setup() function to set the pinMode:

```
void setup() {
  // sets pinmode for LED
  pinMode(BlueledPin, OUTPUT);
}
```

3. Input the following `loop()` function to read values from the microphone to register a clap:

```
void loop() {
  // read the current microphone value
  micVal = analogRead(micPin);
  // micVal of 30 registers a clap
  if (micVal > 30 && !litUp) // turn LED on with one clap
  {
    digitalWrite(BlueledPin, HIGH);
    litUp = true;
  }
  else if (micVal > 30 && litUp) // turn LED off with two claps
  {
    delay(500); // wait half a second between claps to turn off
    micVal = analogRead(micPin);
    if (micVal > 30)
    {
      digitalWrite(BlueledPin, LOW);
      litUp = false;
      delay(1000); // wait a second before being able to turn the light on again
    }
  }
}
```

We set the clap to register at a value greater than 30 which should be adequate, but feel free to adjust this to what works best. Also, note that to register claps to turn off the LED, we wait half a second before detecting the second clap. After turning the LED off, we wait a second before reading the values to turn the LED back on again.

Note: For Rev 2 of the Lenovo Educational Board, your microphone may misread values leading to the LED not turning on when expected

Result

You have created a smart lightbulb that turns on when you clap once and turns off if you clap twice.

Now It's Your Turn!

- Can you make the LED light up to louder sounds like an alarm?
- If you hear a clap, can you play sounds on the buzzer like music? And turn it off as well?

Lesson 1 – 3: Control LEDs With Wifi



OVERVIEW

In this lesson we will:

- Learn how WIFI networks work with Arduino
- Write a program to activate LEDs with buttons on a webpage

Teacher Guide

In this lesson, students will interact with Wi-Fi to turn the LEDs on their board on and off. Before this lesson, we recommend making sure that you know the Wi-Fi network name and password for the students to connect to.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

Wi-Fi for Arduino

There are many options to connect your Arduino board to a WIFI network, but your Educational board is equipped with an ESP32-WROOM-32D processor so no additional equipment needed. ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi. By connecting your Arduino board to a WIFI network, you will be able to remotely control the yellow, blue, and green LEDs on your board anywhere in the house! Keep in mind that your device and the board has to be in the same network to control the lights.

What is HTML?

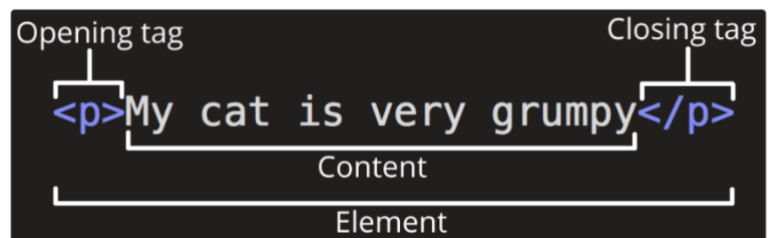
Throughout the lessons that you have completed so far, we have used a programming language called C or C++ to interact with the Arduino IDE. However, in this lesson, we will begin to use the Lenovo Educational Board across the Internet. In order to create documents that are to be displayed in a web browser from an IP address, we will use HTML (Hypertext Markup Language). This programming language is used to achieve font, graphic, color, and hyperlink effects on a web browser. For this lesson, we will begin using HTML (which will be used more in advanced lessons). We will explain



the basics of HTML below, but if you want to learn more please visit the following link:

https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

Now, that you have some context on what HTML is, what does it look like? HTML is divided into sections of elements that have an opening and closing tag for every element. Some common opening tags are <h> for heading and <p> for paragraph. Additionally, HTML commonly uses **hexadecimal** to display color values. This is a quick and easy way to store larger values.



Procedure

In this tutorial, we will move on to something a little bit more advanced! We will write a sketch to control the LEDs over a webpage on your phone or any mobile device.

This sketch runs a web server on the board and serves up HTML to the browser on the client side to allow the user to control the yellow, blue, and green LEDs over the web. As mentioned before, the laptop/smartphone needs to be on the same WIFI network as the Lenovo Educational Board for this to work.

1. In a blank sketch, include the `WiFi.h`, `WebServer.h`, and `Preferences.h` libraries and define the LED pins with the following lines:

```
// include the following libraries
#include <WiFi.h>
#include <WebServer.h>
#include <Preferences.h>
// LED Pins
#define LED_YELLOW 16
#define LED_GREEN 15
#define LED_BLUE 18
```

2. Input the following variables:

```
// WiFi credentials
String ssid = "";
String password = "";
```

4. Next, we will create a web server instance on port 80 with the following line:

```
// Create a web server instance on port 80.
WebServer server(80);
Preferences preferences;
```

** Port 80 is the port number assigned to commonly used internet communication protocol, Hypertext Transfer Protocol (HTTP). It is the port from which a computer sends and receives Web client-based communication and messages from a Web server and is used to send and receive HTML pages or data

5. Now, we create a function called `sendResponse()` to generate the HTML content based on the state of the pins and return that to the client.

```
void sendResponse()
{
  String htmltext = "<!DOCTYPE html><html>\n
  <head><meta name=\"viewport\" content=\"width=device-width,\n
initial-scale=1\">\n
  <link rel=\"icon\" href=\"data:,\">\n
  <style>html { font-family: Helvetica; display: inline-block;\n
margin: 0px auto; text-align: center;}\n
  .button { background-color: #E7E7E7; border: none; color: white;\n
padding: 16px 40px;\n
  text-decoration: none; font-size: 30px; margin: 2px; cursor:\n
pointer;}\n
  .button_y {background-color: #FFFF00;}\n
  .button_g {background-color: #4CAF50;}\n
  .button_b {background-color: #008CBA;}</style></head>\n
<body><h1>LENOVO GTC Dos</h1>\n
<h2>LEDs</h2>";

  // Determine state of Yellow LED.
  if (digitalRead(LED_YELLOW) == HIGH)
  {
    // LED is ON. Provide visual indication and set href to enable turning off.
    htmltext += "<a href=\"/Y/0\"><button class=\"button button_y\">Y</button></a>";
  }
  else
  {
    // LED is OFF. Provide visual indication and set href to enable turning on.
    htmltext += "<a href=\"/Y/1\"><button class=\"button\">Y</button></a>";
  }

  // Determine state of Green LED.
  if (digitalRead(LED_GREEN) == HIGH)
  {
    // LED is ON. Provide visual indication and set href to enable turning off.
    htmltext += "<a href=\"/G/0\"><button class=\"button button_g\">G</button></a>";
  }
  else
  {
    // LED is OFF. Provide visual indication and set href to enable turning on.
    htmltext += "<a href=\"/G/1\"><button class=\"button\">G</button></a>";
  }

  // Determine state of Blue LED.
  if (digitalRead(LED_BLUE) == HIGH)
  {
    // LED is ON. Provide visual indication and set href to enable turning off.
    htmltext += "<a href=\"/B/0\"><button class=\"button button_b\">B</button></a>";
  }
  else
  {
    // LED is OFF. Provide visual indication and set href to enable turning on.
    htmltext += "<a href=\"/B/1\"><button class=\"button\">B</button></a>";
  }

  htmltext += "</body></html>";

  // Return the response to the client.
  server.send(200, "text/html", htmltext);
}
```

6. The next function we will create is the handler for root endpoint, which is the landing page.

```
void handleRoot()
{
  // Call helper function that builds the HTML content
  // and sends it back to the client.
  sendResponse();
}
```

7. Now, we create functions to turn each LED on/off.

```
void handleYellowLed_Off()
{
  // Turn the Yellow LED off.
  digitalWrite(LED_YELLOW, LOW);

  // Send HTTP response.
  sendResponse();
}

// Handler to turn Yellow LED On.
void handleYellowLed_On()
{
  // Turn the Yellow LED on.
  digitalWrite(LED_YELLOW, HIGH);

  // Send HTTP response.
  sendResponse();
}
```

8. Repeat step 7 above and create 2 functions to turn on/off the Green and Blue LED.

9. Input the getWifi() function to read WiFi information from the EEPROM:

```
// get Wifi info
void getWifi()
{
  preferences.begin("credentials", false);
  ssid = preferences.getString("ssid", "");
  password = preferences.getString("password", "");
  if (ssid == "" || password == ""){
    Serial.println("No ssid or password saved.");
  }
  else {
    Serial.println("Successfully read WiFi information.");
  }
}
```

10. Input the following `setup()` function:

```
void setup() {
  // set the baud rate on Serial port
  Serial.begin(115200);
  Serial.println("HPE GTC Dos says Hello!");
  getWifi();

  // Initialize LED pins
  pinMode(LED_YELLOW, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_BLUE, OUTPUT);
  digitalWrite(LED_YELLOW, LOW);
  digitalWrite(LED_GREEN, LOW);
  digitalWrite(LED_BLUE, LOW);

  // Establish WiFi connection
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println("");

  // Initialize and start server
  server.on("/", handleRoot);
  server.on("/Y/0", handleYellowLed_Off);
  server.on("/Y/1", handleYellowLed_On);
  server.on("/G/0", handleGreenLed_Off);
  server.on("/G/1", handleGreenLed_On);
  server.on("/B/0", handleBlueLed_Off);
  server.on("/B/1", handleBlueLed_On);

  server.begin();
  Serial.println("Web server is now running.");
  Serial.println("Type in the IP address above to connect.");
}
```

11. Input the following `loop()` function:

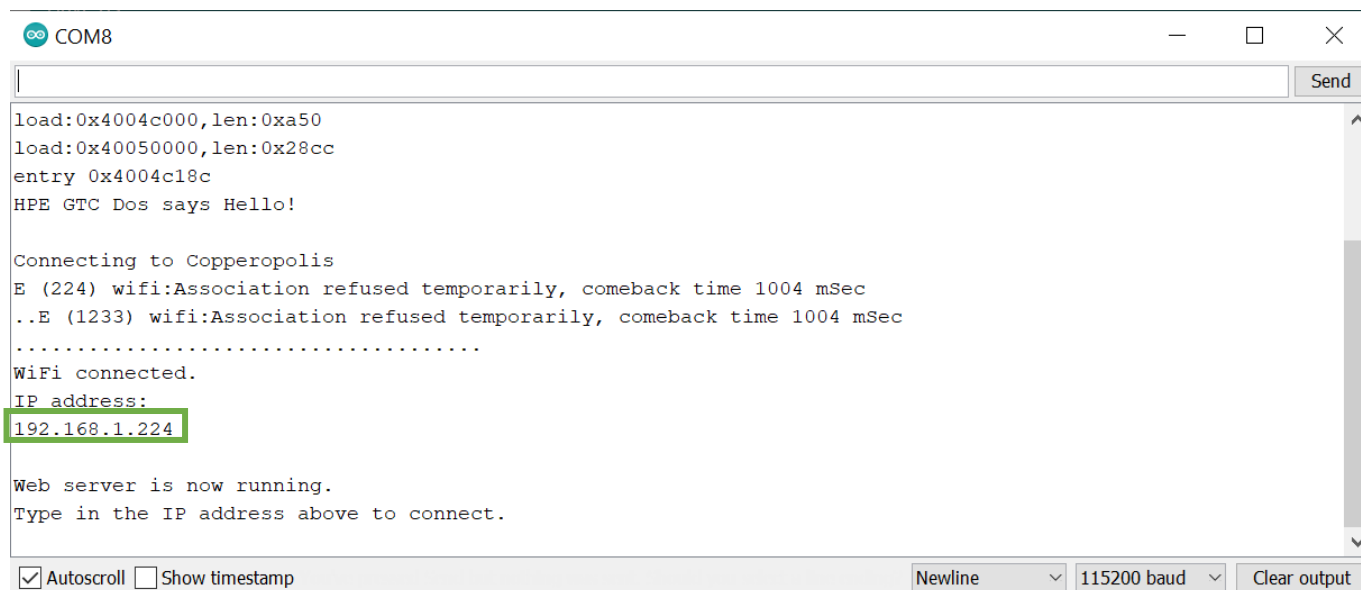
```
void loop() {
  // Start accepting and processing client connections
  server.handleClient();
}
```

Result

After you have created all the additional functions and filled out the **setup()** and **loop()** functions, it's time to execute the program!

1. Connect your board to your computer
2. Set your COM port and open the Serial Monitor
3. Compile and upload your sketch on to your board

You should get the following output from your Serial Monitor



```
COM8
load:0x4004c000,len:0xa50
load:0x40050000,len:0x28cc
entry 0x4004c18c
HPE GTC Dos says Hello!

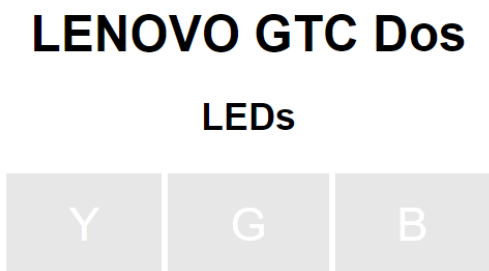
Connecting to Copperopolis
E (224) wifi:Association refused temporarily, comeback time 1004 mSec
..E (1233) wifi:Association refused temporarily, comeback time 1004 mSec
.....
WiFi connected.
IP address:
192.168.1.224

Web server is now running.
Type in the IP address above to connect.
```

Autoscroll Show timestamp Newline 115200 baud Clear output

4. Copy the IP address from the output and go to it on a web browser of your choice

The web page should look like the following image:



An **Internet Protocol address (IP address)** is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication.

- 5. Click on the Y to light up the Yellow LED, the G button to light up the Green LED, and the B button to light up the Blue LED**
- 6. Click on the buttons again to turn of the LEDs**

Now It's Your Turn!

- Can you turn on the Neopixel or the Buzzer using WiFi connections?

Lesson I – 4: Can You Feel the Heat?



OVERVIEW

In this lesson we will:

- Utilize both the temperature sensor and Neopixel to display temperature
- Write a simple program that lights up the Neopixel according to the temperature

Teacher Guide

In this lesson, students will be interacting with the Neopixel and the temperature sensor. If the students have already completed the Neopixel and temperature sensor lessons, then this should come quite easy as most of the code is similar.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

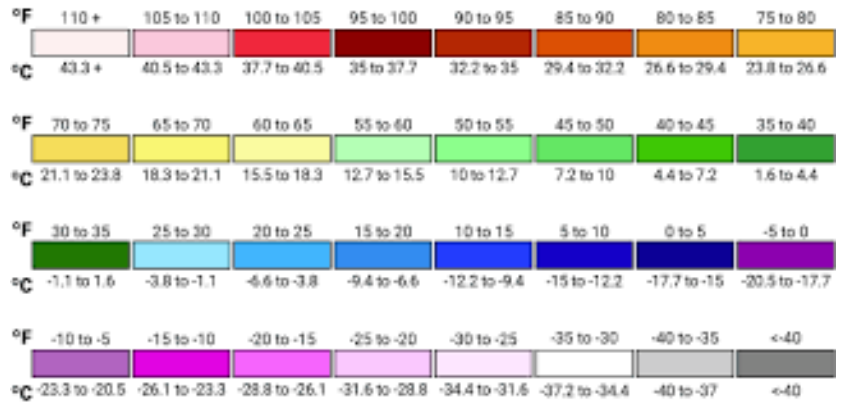
3. PIN number

Did you define the PIN number correctly in your program?

Background Info

Naturally, we associate the color red as hot and the color blue as cold. Think of a mood ring or anything similar; the reason that they change colors is because of the temperature and not actually your emotions. A mood ring is a ring that contains a thermochromic element, such as liquid crystal, that changes colors based upon the temperature of the finger of the wearer. Another product you might know

that changes colors when the temperature changes is color changing nail polish. Color changing nail polish works that same way as the mood ring. An elevation in body temperature, for example: running your hands under warm/hot water, or going outside in hot weather will cause the nail polish color to change.



So what if we create a program that can change the color of the Neopixel LED on the Arduino board according to what the current temperature is? In order to achieve this, we would need to interact with two devices on our board, the Temperature Sensor and the Neopixel LED.

Procedure

Do you still remember how to change the color on the Neopixel or get the current temperature in Fahrenheit from the temperature sensor? If you do, stop reading this and try it on your own first! If you don't, that's okay, we will walk you through it.

1. First, include the required libraries:

```
#include <Wire.h> // I2C library
#include "ens210.h" // ENS210 library
#include <Adafruit_NeoPixel.h> // Neopixel library
```

2. Create and define variables and pins:

```
#define NEOPIXEL_PIN 1 // Neopixel is on pin 1
#define NEOPIXEL_COUNT 1 // There is only 1 RGB LED on Dos.
#define NEOPIXEL_INDEX 0 // RGB LED is at index 0.
#define DELAYVAL 500 // Time (in milliseconds) to pause between pixels
float CurTemp; // store current temperature
```

3. Create Adafruit_NeoPixel and ENS210 objects:

```
Adafruit_NeoPixel Pixel(NEOPIXEL_COUNT, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);
ENS210 ens210;
```

4. Input the following `setup()` function to initialize the Neopixel and Temperature sensor:

```
void setup() {
  // set the baud rate on the serial port
  Serial.begin(115200);

  // initialize the Neopixel object
  Pixel.begin();
  // set all pixel colors to OFF
  Pixel.clear();
  Pixel.show();

  // initialize I2C
  Wire.begin();
  // Enable ENS210
  ens210.begin();
}
```

5. Input the following inside the `loop()` function to read the temperature values:

```
void loop() {
  // resets Neopixel color
  Pixel.setPixelColor(NEOPIXEL_INDEX, 0, 0, 0);
  // reads temperature data to CurTemp variable
  int t_data, t_status, h_data, h_status;
  ens210.measure(&t_data, &t_status, &h_data, &h_status);
  CurTemp = ens210.toFahrenheit(t_data, 10) / 10.0;
  Serial.print("Temp: ");
  Serial.print(CurTemp);
  Serial.print(" F");
  Serial.println();
}
```

6. Inside the `loop()` function add the temperature thresholds to change the Neopixel color:

```
// temperatures are associated with a color
if (CurTemp <= 35.0) // set to blue
{
  Pixel.setPixelColor(NEOPIXEL_INDEX, 0, 0, 255);
}
else if (CurTemp > 35.0 && CurTemp <= 50.0) // set to light blue
{
  Pixel.setPixelColor(NEOPIXEL_INDEX, 30, 30, 255);
}
else if (CurTemp > 50.0 && CurTemp <= 60.0) // set to green
{
  Pixel.setPixelColor(NEOPIXEL_INDEX, 0, 255, 0);
}
else if (CurTemp > 60.0 && CurTemp <= 70.0) // set to yellow
{
  Pixel.setPixelColor(NEOPIXEL_INDEX, 255, 255, 0);
}
else if (CurTemp > 70.0 && CurTemp <= 80.0) // set to orange
{
  Pixel.setPixelColor(NEOPIXEL_INDEX, 140, 30, 0);
}
else if (CurTemp > 80.0 && CurTemp <= 90.0) // set to red
{
  Pixel.setPixelColor(NEOPIXEL_INDEX, 255, 15, 10);
}
else if (CurTemp > 90.0 && CurTemp <= 100.0) // set to pink red
{
  Pixel.setPixelColor(NEOPIXEL_INDEX, 255, 50, 50);
}
else if (CurTemp > 100.0) // set to white
{
  Pixel.setPixelColor(NEOPIXEL_INDEX, 255, 255, 255);
}
// push the color to the neopixel
Pixel.show();
// waits 5 seconds
delay(5000);
}
```

Result

After you have completed the procedure, now you should be able to upload the program onto the board. Opening the serial monitor is optional if you want to know the exact temperature, as well as to check if the Neopixel displays the correct color according to the temperature.

Another tip for testing the program is to blow hot air or put your hand over the temperature sensor to increase the temperature. Then check to see if the color on the Neopixel changes or not.

Now It's Your Turn!

- Can you display the temperature value to an OLED display and change the LEDs based on the humidity?

Lesson 1 – 5: Morse Code



OVERVIEW

In this lesson we will:

- Learn how Morse code works
- Take a string and output Morse code to an LED
- Input Morse code through the buttons and convert it into strings to display on the Serial Monitor

Teacher Guide

In this lesson, students will interact with morse code. This exercise is long but will be rewarding once the students finish. Before starting this lesson, we recommend looking at the morse code alphabet and learning “SOS” as this is a very easy and common morse code signal. Additionally, students will input their values via a button to the Serial to create letters and once a certain button is pressed, display to a LED.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

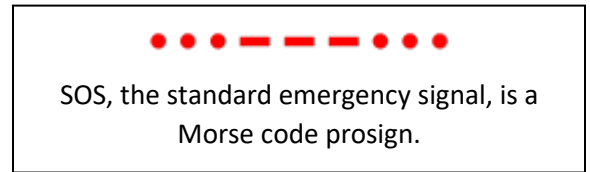
Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

Morse code is a character encoding scheme used in telecommunication that encodes text characters as standardized sequences of two different signal durations called *dots* and *dashes* or *dits* and *dahs*. Morse code is named for Samuel F. B. Morse, an inventor of the telegraph.



The International Morse Code encodes the 26 English letters A through Z, some non-English letters, the Arabic numerals and a small set of punctuation and procedural signals (prosigns). There is no distinction between upper and lower-case letters.

Each Morse code symbol is formed by a sequence of dots and dashes. The dot duration is the basic unit of time measurement in Morse code transmission. The duration of a dash is three times the duration of a dot. Each dot or dash within a character is followed by period of signal absence, called a *space*, equal to the dot duration. The letters of a word are separated by a space of duration equal to three dots, and the words are separated by a space equal to seven dots. To increase the efficiency of encoding, Morse code was designed so that the length of each symbol is approximately inverse to the frequency of occurrence in text of the English language character that it represents. Thus the most common letter in English, the letter "E", has the shortest code: a single dot. Because the Morse code elements are specified by proportion rather than specific time durations, the code is usually transmitted at the highest rate that the receiver is capable of decoding. The Morse code transmission rate (*speed*) is specified in *groups per minute*, commonly referred to as *words per minute*.

Morse code is usually transmitted by on-off keying of an information carrying medium such as electric current, radio waves, visible light or sound waves. The current or wave is present during time period of the dot or dash and absent during the time between dots and dashes.

Morse code can be memorized, and Morse code signaling in a form perceptible to the human senses, such as sound waves or visible light, can be directly interpreted by persons trained in the skill.

Because many non-English natural languages use other than the 26 Roman letters, Morse alphabets have been developed for those languages.

In an emergency, Morse code can be generated by improvised methods such as turning a light on and off, tapping on an object or sounding a horn or whistle, making it one of the simplest and most versatile methods of telecommunication. The most common distress signal is SOS – three dots, three dashes, and three dots – internationally recognized by treaty.

International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A ● —
B — ● ● ●
C — ● — ●
D — ● ●
E ●
F ● ● — ●
G — — ●
H ● ● ● ●
I ● ●
J ● — — —
K — ● —
L ● — ● ●
M — —
N — ●
O — — —
P ● — — ●
Q — — ● —
R ● — ●
S ● ● ●
T —

U ● ● —
V ● ● ● —
W ● — —
X — ● ● —
Y — ● — —
Z — — ● ●

1 ● — — —
2 ● ● — —
3 ● ● ● — —
4 ● ● ● ● —
5 ● ● ● ● ●
6 — ● ● ● ●
7 — — ● ● ●
8 — — — ● ●
9 — — — — ●
0 — — — — —

4. Input the following `loop()` function:

```
void loop() {
  if (complete) // outputs the morse code to LED
  {
    complete = false;
    String morseWord = encode(finalMsg);
    for(int i = 0; i <= morseWord.length(); i++)
    {
      switch(morseWord[i])
      {
        case '.': // dit
          digitalWrite(PIN_OUT, HIGH);
          delay(UNIT_LENGTH);
          digitalWrite(PIN_OUT, LOW);
          delay(UNIT_LENGTH);

          break;

        case '-': // dah
          digitalWrite(PIN_OUT, HIGH);
          delay(UNIT_LENGTH * 3);
          digitalWrite(PIN_OUT, LOW);
          delay(UNIT_LENGTH);

          break;

        case ' ': // gap
          delay(UNIT_LENGTH);
      }
    }
    digitalWrite(16, HIGH);
    delay(UNIT_LENGTH);
    digitalWrite(16, LOW);
    delay(UNIT_LENGTH * 3);
    finalMsg = "";
    Serial.println();
  }
  else // takes in morse code from button
  {
    while (digitalRead(buttonTwo) != HIGH)
    {
      String myMsg = GetLetter();
      Translate(myMsg);
    }
    complete = true;
  }
}
```

First, the complete Boolean is reset each loop and we read whether the button to output to the LED is pushed and change the Boolean accordingly. If the Boolean is true, we run the if statement which is explained below.

The String morseWord is just our desired message encoded in Morse Code. This is done by feeding the message (in this case it is the variable, finalMsg) to a method called encode, which is written outside of the loop method. We will detail below how encode works. Please input the encode function below and outside of the loop function.

```
// translates letters to morse code
String encode(String string)
{
    size_t i, j;
    String morseWord = "";
    for( i = 0; string[i]; i++ )
    {
        for( j = 0; j < 47; j++ )
        {
            if( string[i] == codex[j] )
            {
                morseWord += dots_and_dashes[j];
                break;
            }
        }
        morseWord += " "; // Add tailingspace to separate the chars
    }
    return morseWord;
}
```

Since strings are just arrays of characters, we can for loop through the string and read each character individually at index i. Then we have a nested for loop inside that checks through the codex variable to find the character that matches. If it matches, then we add the corresponding code to morseWord from dots_and_dashes. We add a small space after each character so the LED can briefly refresh before starting to display the next letter. Finally, once we have found the entire encoding for the input string, we return morseWord.

Now we will take a closer look at the switch case statements that you already inputted into the loop. Please note that you do NOT need to type the following information again.

```
switch(morseWord[i])
{
    case '.': // dit
        digitalWrite(PIN_OUT, HIGH);
        delay(UNIT_LENGTH);
        digitalWrite(PIN_OUT, LOW);
        delay(UNIT_LENGTH);
        break;
    case '-': // dah
        digitalWrite(PIN_OUT, HIGH);
        delay(UNIT_LENGTH * 3);
        digitalWrite(PIN_OUT, LOW);
        delay(UNIT_LENGTH);
        break;
    case ' ': // gap
        delay(UNIT_LENGTH);
}
```

Back to the loop method. Once we are given morseWord, we again write a for loop to read all of its characters in order. Here we utilize **switch and case statements**, which are an excellent tool for pattern matching if we want to quickly and cleanly organize code to recognize variations of an input.

In this case, we want to read in a character (this is what's in the parentheses, switch(morseWord[i]) and case on it. We only have to account for 3 cases because Morse code only involves 3 types of output: dits, dahs, and gaps. Case statements need a **break;** statement afterwards because if that break was not there, they would continue to read the cases underneath them and run their code as well. For example, if there were no breaks after the first two cases and we read in a dit, the code for dit, dah, and gap would all run! Once our code determines which case describes the current character at index i in morseWord, it will run the appropriate code, turning on PIN_OUT when appropriate and for however much time is necessary, in relation to UNIT_LENGTH.

At the end of each iteration of loop, when our message has fully been relayed, we briefly reset the orange LED to show that the message has been completed and will restart shortly.

Now, we look to the else statement in the loop function. This contains two functions inside of it to get input from the user using morse code. We will explain these two functions below:

The GetLetter function will return a string of dots and dashes.

```
// reads input from button to morse code
String GetLetter()
{
    unsigned long START, STOP, TIME;
    String myMsg = "";
    while(true)
    {
        while(digitalRead(BUTTON) == LOW)
        {
            // do nothing until button is pressed
        }
        digitalWrite(PIN_OUT, HIGH);
        START = millis();
        while(digitalRead(BUTTON) == HIGH)
        {
            // do not register input until button is released
        }
        STOP = millis();
        digitalWrite(PIN_OUT, LOW);
        TIME = STOP - START;
    }
}
```


This function will translate the string from the previous function and display it to the serial monitor by taking input from the user and concatenating a string with dots and dashes based on how long the input button is pressed and held for.

The function starts by entering a while loop with the “true” condition: this way, we can add multiple dots or dashes without having to use more logic in the “loop()” function. The two empty, `while` loops within this function are used to calculate the amount of time that passes between pressing and releasing the button. The first loop will not end until the button is pressed, and the second loop will not end until the button is released.

The `millis()` function returns the amount of time that has passed since the current program began. By calling this function when the button is pressed, and then once more when it is released, then we can determine how long the button is held for. We also turn on an LED while the button is pressed down.

We now continue to input code into the `GetLetter()` function where we left off.

```
if (TIME > 25)
{
  myMsg += Dot_or_Dash(TIME);
}
while((millis() - STOP) < 750)
{
  delay(10);
  if (digitalRead(BUTTON) == HIGH)
  {
    break;
  }
}
if ((millis() - STOP) >= 750)
{
  return myMsg;
}
}
```

If the button was only pressed down for a fraction of a second (less than 25ms), it is more likely to have been the switch bouncing, so we will invalidate that press. Otherwise, we will take that input and add either a dot or a dash to our string. If 0.750 seconds go by without another button press, then whatever is stored in that string will be sent to the translator (i.e. it marks the end of a letter).

Next, we explore the `Dot_or_Dash()` function which was used inside of the `GetLetter()` function.

```

// determine if it is a dot or dash
char Dot_or_Dash (unsigned long TIME)
{
    char DoD;
    if (TIME < 500)
    {
        DoD = '.';
    }
    else
    {
        DoD = '-';
    }
    return DoD;
}

```

If the button press was less than 0.500 seconds, then it is a dot. Otherwise, the press is a dash. Feel free to customize this to your own liking.

Now, we will input the translate function which converts morse code to letters.

```

// translates morse code to letters
void Translate (String myMsg)
{
    for (int i = 0; i < 47; i++)
    {
        if (myMsg == dots_and_dashes[i])
        {
            Serial.print(codex[i]);
            finalMsg += codex[i];
            return ;
        }
    }
    Serial.print("ERROR, I cannot understand you!\n");
    return ;
}

```

The above code takes the string produced by GetLetter() and runs through our codex to see if that string matches any of our predefined strings. If it does, then it prints that letter to the serial monitor. If it doesn't, then we print an Error.

Finally, we have reached the end of our loop function which has a 1.5 second delay. This requires waiting 1.5 seconds between inputting characters in morse code into the button. Additionally, to output your string to morse code using the LEDs, as soon as the letter displays on the screen hold button2 for 2 seconds. This delay can be changed according to preference.

Result

You have now learned about how Morse code works and how to output it on your Arduino board! You have also learned about switch and case statements and how arrays work.

Now It's Your Turn!

- Can you make the input message read from typing in the Serial Monitor?
- Can you light up the Neopixel instead of a LED for Morse code?
- Can you add a break for words? Will it be a certain string of characters or will it be after a certain amount of time has passed?
- Can you send a message with your encoder via email?

Lesson I – 6: Proximity Sensor



© izmo cars

OVERVIEW

In this lesson we will:

- Utilize proximity sensor and buzzer to act like a car's back up camera

Teacher Guide

In this lesson, the students will be interacting with the buzzer and gesture sensor to detect distance from the board. The motion sensor starts to pick up an object about 18 inches away from the sensor and reads until an object is about 2 inches away. Please note, that even when the program is running and no object is detected, the buzzer will still sound which may get annoying.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

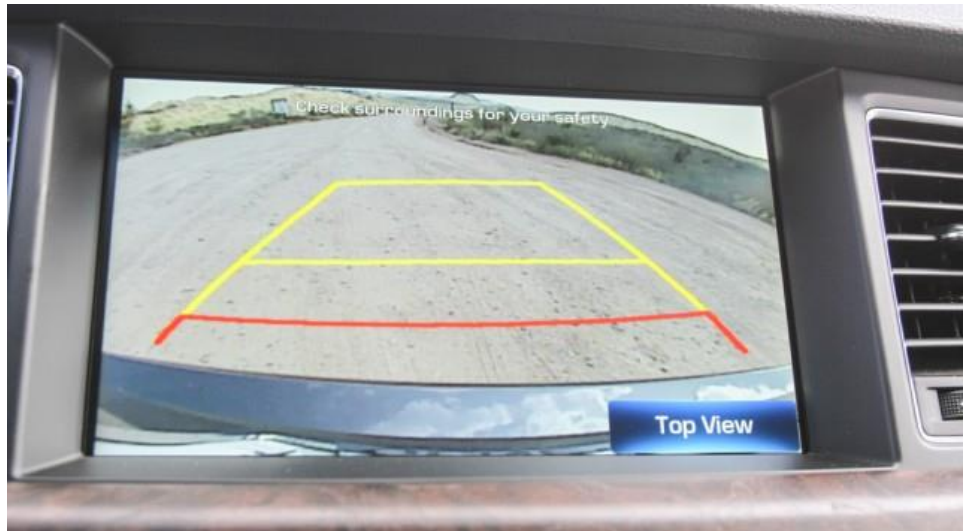
Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

What is a proximity sensor you might ask? This device measure how close or far that objects are away from it. These types of sensors are then able to do certain actions within a given distance threshold. For example, when a car is backing up and you hear beeps increasing in frequency as you get closer to an object to signal how near you are. Additionally, many phones use a proximity sensor when you pick up a phone call to dim or turn off the screen if the phone is held up to your ear. The applications for this device is endless, but for this lesson we will use the sensor to resemble the back up camera on a car.



Procedure

1. Start by including the following libraries:

```
// include the following libraries
#include <Wire.h>
#include <SparkFun_APDS9960.h>
```

2. Create a SparkFun_APDS9960 object.

```
// object creation
SparkFun_APDS9960 apds = SparkFun_APDS9960();
```

3. Include the following constants and global variables:

```
uint8_t proximity_data = 0; // proximity value
const float midC = 256.0; // the hz value of middle C
const int piezoPin = 11; // the number of the buzzer pin
```

4. Input the following `setup()` function to set the baud rate, set the pinModes, and initialize the gesture sensor:

```
void setup() {
  // Initialize Serial port
  Serial.begin(9600);
  pinMode(piezoPin, OUTPUT);
  // Initialize gesture sensor for proximity
  apds.init();
  apds.enableProximitySensor(false);
  apds.setProximityGain(PGAIN_2X);
}
```

5. Input the following `loop()` function to read the proximity values:

```
void loop() {
  // Read the proximity value
  apds.readProximity(proximity_data);
}
```

6. Inside of the loop function that you inputted above include:

```
// each test case is in a loop if proximity is within a threshold
while(proximity_data < 15)
{
  myTone(piezoPin, midC, 250);
  delay(1100);
  apds.readProximity(proximity_data);
}
while(proximity_data >= 15 && proximity_data < 25)
{
  myTone(piezoPin, midC, 250);
  delay(900);
  apds.readProximity(proximity_data);
}
while(proximity_data >= 25 && proximity_data < 50)
{
  myTone(piezoPin, midC, 250);
  delay(700);
  apds.readProximity(proximity_data);
}
while(proximity_data >= 50 && proximity_data < 100)
{
  myTone(piezoPin, midC, 250);
  delay(500);
  apds.readProximity(proximity_data);
}
while(proximity_data >= 100 && proximity_data < 150)
{
  myTone(piezoPin, midC, 250);
  delay(300);
  apds.readProximity(proximity_data);
}
while(proximity_data >= 150 && proximity_data < 200)
{
  myTone(piezoPin, midC, 250);
  delay(200);
  apds.readProximity(proximity_data);
}
while(proximity_data >= 200)
{
  myTone(piezoPin, midC, 250);
  delay(100);
  apds.readProximity(proximity_data);
}
}
```

These are the test cases for distances away from the sensor so when you are within a certain range, the beeping will have shorter delays as you get closer to the sensor.

7. Input the myTone method to play the sound from the buzzer:

```
// plays the tone we want by using the vibrations of the buzzer
void myTone(byte pin, uint16_t frequency, uint16_t duration)
{ // input parameters: Arduino pin number, frequency in Hz, duration in milliseconds
  unsigned long startTime=millis();
  unsigned long halfPeriod= 1000000L/frequency/2;
  while (millis()-startTime< duration)
  {
    digitalWrite(pin,HIGH);
    delayMicroseconds(halfPeriod);
    digitalWrite(pin,LOW);
    delayMicroseconds(halfPeriod);
  }
}
```

Result

Now you can hold your hand starting 18 inches away from the sensor the beeping will increase in frequency as you get closer to the sensor and then slower again as you get further away.

Now It's Your Turn!

- Can you make the Neopixel turn red if you get within two inches of the sensor?

Lesson I – 7: Mini Sound Board



OVERVIEW

In this lesson we will:

- Write a program to convert button inputs into sounds

Teacher Guide

Students will be using buttons on the board to play sounds. Please remind students to be cognizant to others because playing lots of sounds may become overwhelming or annoying for other students. This lesson uses an array to store values, but we learned about these during the morse code lesson.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

A soundboard is a computer program, Web application, or device, traditionally created in Adobe Flash that catalogues and plays many short soundbites and audio clips. Soundboards are self-contained, requiring no outside media player. In recent years soundboards have been made available in the form of mobile apps available on iPhone App Store and Google Play. In response to Adobe and web browser developers deprecating support for Flash, HTML5-based soundboards are now gaining popularity in recent years.



Procedure

1. Define some constants that represent the pin numbers of the buzzer, LEDs, and buttons which we'll be using.

```
// define button, LED, and buzzer pins
#define BUZZER 11
#define YELLOW 16
#define GREEN 15
#define BLUE 18
#define Button1 4
#define Button2 5
#define Button3 6
#define MODE 7
```

2. Declare the following arrays:

```
// create the notes to play
int myChords [][][3] = {
  {440, 555, 660}, // A Major
  {523, 660, 784}, // C Major
  {392, 494, 588}, // G Major
};

int myLED[3] = {YELLOW, GREEN, BLUE}; // Yellow = A Major,
// Green = C Major, Blue = G Major
```

Here, we've declared two more arrays: one to store our chords with the frequencies of each note, and another to store the LED displays to match up with the chord.

3. Create global variables to store the states of the buttons.

(i.e. pressed or not-pressed)

```
// button states and modes
int mode_select = 0;
int Button1_State = 0;
int Button2_State = 0;
int Button3_State = 0;
int MODE_State = 0;
```

4. Input the following `setup()` function:

```
void setup() {
  Serial.begin(115200);
  // set the modes for buttons, LEDs, and buzzer
  pinMode(YELLOW, OUTPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(BLUE, OUTPUT);
  pinMode(MODE, INPUT);
  pinMode(Button1, INPUT);
  pinMode(Button2, INPUT);
  pinMode(Button3, INPUT);
  pinMode(BUZZER, OUTPUT);

  // initialize the LED display
  int i = mode_select % 3;
  digitalWrite(myLED[i], HIGH);
}
```

Here we are just setting up our pins to be inputs or outputs, and we turn on the first LED in our sequence.

5. Input the following `loop()` function:

```
void loop() {
  // determine chord to play based on mode
  int n = mode_select % 3;
  int note1 = myChords[n][0];
  int note2 = myChords[n][1];
  int note3 = myChords[n][2];
  MODE_State = digitalRead(MODE);
```

The first thing to happen in each loop is to set the notes. Let's break down that array referencing. If we keep toggling our Mode, then we can bound it between the values of 0 and 2 by taking the modulus of our counter with the number 3. Then, we select that n^{th} triad from our array and set the first, second, and third notes to our local variables.

6. Set up the following `if` statement following the code inputted above in the `loop()` function:

```
// play different notes based off mode selected by button at pin 4
if (MODE_State) // then change the current LED and the cords stored
{
  delay(250);
  mode_select++;
  digitalWrite(myLED[n], LOW);
  n = mode_select % 3;
  digitalWrite(myLED[n], HIGH);
}
// play different note in cord based on button number
if (digitalRead(Button1) == HIGH)
{
  myTone(BUZZER, note1, 250);
}
if (digitalRead(Button2) == HIGH)
{
  myTone(BUZZER, note2, 250);
}
if (digitalRead(Button3) == HIGH)
{
  myTone(BUZZER, note3, 250);
}
return;
}
```

The first statement increments our Mode (we add in a delay so that you have a buffer to lift-up without registering as another press). We turn off the current LED and then turn on the new LED.

Then, depending on which button is pressed, we play one of the three notes in our triad.

7. Finally, input the `myTone` method to play the sound from the buzzer below the `loop()` function.

```
// plays the tone we want by using the vibrations of the buzzer
void myTone(byte pin, uint16_t frequency, uint16_t duration)
{ // input parameters: Arduino pin number, frequency in Hz, duration in milliseconds
  unsigned long startTime=millis();
  unsigned long halfPeriod= 1000000L/frequency/2;
  while (millis()-startTime< duration)
  {
    digitalWrite(pin,HIGH);
    delayMicroseconds(halfPeriod);
    digitalWrite(pin,LOW);
    delayMicroseconds(halfPeriod);
  }
}
```

Result

Now we have a simple sound board where a single button will play one note in a triad, and we can even select which triad is being played.

Now It's Your Turn!

- Can you set up a single button press to play a small melody?
- Can you string together button presses and melodies to create a song?
- Can you combine this with the WiFi project to create an online interface for the sounds?

Lesson I – 8: Motion Detector



OVERVIEW

In this lesson we will:

- Learn about what a gesture sensor is and how they work
- Create a project using the gesture sensor on the Lenovo Educational Board

Teacher Guide

In this lesson, the students will be interacting with the motion sensor to detect movement. The main forms of movement are up, down, left, and right. Please note that sometimes gestures may be misread. Also, we recommend moving your hand about 6 inches away at a relatively normal speed for the best results. If this does not work, try moving two fingers in various directions about 1 inch from the sensor.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

The first motion detector was invented by Samuel Bango to detect burglar or thieves. Nowadays, motion sensors are used everywhere, such as security systems in your home, banks, shopping malls, etc. Retail or grocery stores have motion detectors at each entrance, so whenever you come near the door, the door will automatically open. A lot of modern cars are also equipped with motion sensors to detect nearby cars and sense how close they are to your car. This feature in cars has prevented serious accidents from happening on the road.

Motion detectors detect humans or moving objects motions and output data to a controller. There are different types of motion detectors. Commonly, motion detectors use different types of sensors to detect movements such as passive infrared sensors, ultrasonic sensors, and microwave sensors.

Passive infrared sensors detect a person's body heat when they come close to the sensor and produce digital outputs. PIR sensors are small, low power, affordable and easy to use.

Ultrasonic sensors are used to measure a moving object's reflection. When an object enters the spectrum of the ultrasonic sensor, the sound waves get reflected back creating echoes, and so this generates electric pulse. Therefore, the ultrasonic sensors detect motion with these echo patterns. The EDU board is equipped with a motion sensor, Adafruit_APDS9960, so when the sensor detects a movement, it will send signals to the micro-controller. You can then program the board to simulate different actions based on the output of the sensor.

You can visit <https://www.elprocus.com/motion-detector-circuit-with-working-description-and-its-applications/> for more information about motion sensor.

Procedure

In this lesson, we will build a simple project utilizing the motion sensor to detect directions of gestures.

1. First, we will include the appropriate libraries:

```
// include Melopero library
#include "Melopero_APDS9960.h"
#include <Wire.h>
#include <SparkFun_APDS9960.h>
```

Please note that you must install the Melopero_APDS9960 library from the library manager.

2. Now, we will create the SparkFun and Melopero objects:

```
// create objects
SparkFun_APDS9960 apds = SparkFun_APDS9960();
Melopero_APDS9960 device;
```

3. Input the following `setup()` function to initialize and reset the gesture sensor:

```
void setup() {
  Serial.begin(9600); // set serial to 9600
  // set up the gesture sensor
  Wire.begin();
  device.initI2C(0x39, Wire);
  device.reset(); // Reset the device
  device.resetGestureEngineInterruptSettings();
  apds.init();
  Serial.println("Device initialized correctly!");
}
```

4. Continue to input the following in the `setup()` function to set the gesture sensor settings:

```
// Gesture settings to set the distance for reading gestures
device.enableGesturesEngine();
device.setGestureProxEnterThreshold(25);
device.setGestureExitThreshold(20);
device.setGestureExitPersistence(EXIT_AFTER_4_GESTURE_END);

// turn on gesture sensor
device.wakeUp();
}
```

5. Input the following `loop()` function to read new gestures from the sensor and print them to the Serial.

```
void loop() {
  // update the status to see if there is a gesture
  device.updateGestureStatus();
  // if gesture read the gesture
  if (device.gestureFifoHasData){
    device.parseGesture(300);

    if (device.parsedUpDownGesture != NO_GESTURE ||
        device.parsedLeftRightGesture != NO_GESTURE)
      Serial.print("Gesture : ");

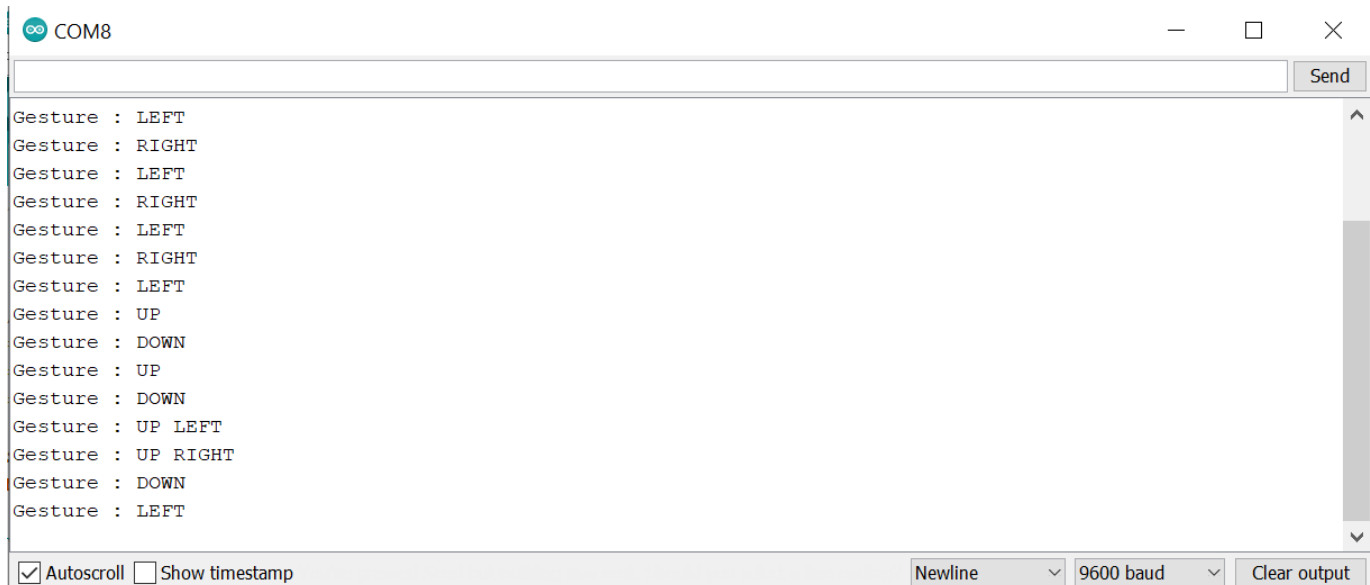
    if (device.parsedUpDownGesture == UP_GESTURE)
      Serial.print("UP ");
    else if (device.parsedUpDownGesture == DOWN_GESTURE)
      Serial.print("DOWN ");

    if (device.parsedLeftRightGesture == LEFT_GESTURE)
      Serial.print("LEFT ");
    else if (device.parsedLeftRightGesture == RIGHT_GESTURE)
      Serial.print("RIGHT ");

    if (device.parsedUpDownGesture != NO_GESTURE ||
        device.parsedLeftRightGesture != NO_GESTURE)
      Serial.println();
  }
}
```

Result

Once you have successfully compiled and uploaded the sketch on to the board, go ahead and open your Serial Monitor to see the output. Wave your hand in front of the sensor in one direction, then again in a different direction. Your Serial Monitor output should look something like the figure below. Please be sure to set the baud rate to 9600.



```
COM8
Send
Gesture : LEFT
Gesture : RIGHT
Gesture : LEFT
Gesture : RIGHT
Gesture : LEFT
Gesture : RIGHT
Gesture : LEFT
Gesture : UP
Gesture : DOWN
Gesture : UP
Gesture : DOWN
Gesture : UP LEFT
Gesture : UP RIGHT
Gesture : DOWN
Gesture : LEFT
```

Autoscroll Show timestamp Newline 9600 baud Clear output

Now It's Your Turn!

- Can you create a sketch that prints out a random even number when you swipe up and a random odd number when you swipe down?

Lesson I – 9: Guess the Number



OVERVIEW

In this lesson we will:

- Use a random number generator
- Write and read values to and from EEPROM
- Use the Serial to communicate guesses and results

Teacher Guide

For this lesson, students will be working with random numbers, the Serial Monitor, and EEPROM to create a small guessing game. Since EEPROM has limited uses (about 100,000) we recommend running this program only a handful of times to prevent hardware damage.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

What is random number? Well, in programming, these numbers are not actually random, instead they are pseudo-random. What this means is that these are calculated using mathematical ways but are still considered random numbers. Additionally, these numbers are oftentimes in a sequence of values where the seed is where in the random sequence you want to start. For this lesson, we will use the **random()** function to get a pseudo-random number from the computer. However, we could use the **randomSeed()** function to get the same values read from the computer (i.e., your first value is a one then mine would also be one if we had the same random seed).

Procedure

1. Include the EEPROM library:

```
#include "EEPROM.h"
```

2. Above the setup() function type the following variables:

```
uint32_t randComputer = 0; // random number selected by computer
uint32_t randGuess = 0; // your guess
int bootCount = 0; // keeps track of number of loops
```

3. Include the following EEPROMClass objects:

```
// Create EEPROM classes
EEPROMClass RANDCOMP("eeprom0");
EEPROMClass GUESS("eeprom1");
```

4. Input the following setup() function:

```
void setup() {
  // set baud rate
  Serial.begin(115200);
  delay(1000);
  // begin EEPROM class
  RANDCOMP.begin(0x500);
  GUESS.begin(0x200);
  Serial.println("Please input a number 1, 2, or 3");
}
```

Here we begin the EEPROM objects and tell the computer where to store the values in the EEPROM.

5. Input the following loop() function:

```
void loop() {
  // run while serial is available
  while(Serial.available() > 0)
  {
    randGuess = Serial.parseInt();
    // A newline character indicates end of input
    if(Serial.read() == '\n')
    {
      //Increment boot number
      ++bootCount;
      // Adjust the values if out of range
      randGuess = constrain(randGuess, 1, 3);
      // select random number in range 1-3
    }
  }
}
```

We read the input from the Serial and convert it to an integer that is constrained to the range 1 through 3. Next, we set the computer number to random number between 1 and 3.

6. Continue inputting the loop() function:

```
//Write: Variables ---> EEPROM stores
RANDCOMP.put(0, randComputer);
GUESS.put(0, randGuess);
Serial.println("Writing to EEPROM:");
Serial.print("Your Guess: ");    Serial.println(randGuess);
Serial.println("-----\n");

// Clear variables
randComputer = 0;
randGuess = 0;
Serial.println("Clearing variables");
Serial.print("Computer Number: ");    Serial.println(randComputer);
Serial.print("Your Guess: ");    Serial.println(randGuess);
Serial.println("-----\n");
delay(3000); // wait 3 seconds before displaying computer number

// Read: Variables <--- EEPROM stores
RANDCOMP.get(0, randComputer);
GUESS.get(0, randGuess);
Serial.println("Reading from EEPROM:");
Serial.print("Computer Number: ");    Serial.println(randComputer);
Serial.print("Your Guess: ");    Serial.println(randGuess);
Serial.println("-----\n");
Serial.println();
```

This is where we write the variables to the EEPROM and then read them after a slight delay to make sure that the number they chose is correct.

7. Finish inputting the loop() function:

```
    if (randComputer == randGuess) // if guessed correctly
    {
        Serial.println("You have guess correctly!");
        delay(0xFFFFFFFF); // max delay value to prevent looping
    }
    else if (bootCount >= 5) // if you ran out of attempts
    {
        Serial.println("Sorry you have run out of attempts.");
        delay(0xFFFFFFFF); // max delay value to prevent looping
    }
    else // print selection statement before looping again
    {
        Serial.println("Please input a number 1, 2, or 3");
    }
}
}
```

Here there are if statements on ending the loop if there have been 5 or more loops or you have guessed the computer's number correctly.

Result

Now we have a fully functioning random number guessing game where you have 5 attempts to guess a number that the computer is thinking of.

Now It's Your Turn!

- Can you change the number range that people are able to guess?
- Can you use EEPROM to make another small game like a number guesser?

Lesson I – 10: Create Your Own Timer



OVERVIEW

In this lesson we will:

- Write a simple program to set buttons to turn on/off a stopwatch and output the total time in the Serial Monitor
- Set a time from the Serial Monitor and make the buzzer go off when the time has elapsed

Teacher Guide

For this lesson, students will be working with the Serial Monitor to create a timer and stopwatch. Please note when typing in the large integers used for division to type the correct number of zeroes or the calculations will be thrown off, leading to negative seconds or minutes being displayed.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

If you have taken any test or played any sports then you have used a timer to track your time. You probably have either used a timer to count down a specific time interval, or a stopwatch to count upwards from zero for measuring the elapsed time. Timers and stopwatches are used everywhere around us, and they are included in most modern electronic devices such as phones, computers, or smart appliances.

We can create both a timer and a stopwatch from the Educational Arduino board by utilizing the buttons and the buzzer. We will output the time through the Serial Monitor.

Procedure

1. First, define the following global variables and pins:

```
// define global variables and pins
#define BUZZER 11
#define Button1 4
#define Button2 5
int elapsedMillis = 0;
int currentMillis = 0;
int startMillis = 0;
int h = 0; int m = 0; int s = 0;
```

2. Input the following `setup()` function to set the pinModes and print the option of timer or stopwatch to the Serial:

```
void setup() {
  // set baud rate
  Serial.begin(9600);
  // set the appropriate state for buzzer and buttons
  pinMode(BUZZER, OUTPUT);
  pinMode(Button1, INPUT);
  pinMode(Button2, INPUT);
  Serial.println();
  Serial.println();
  Serial.println("Please decide whether you want to use the timer or the stopwatch.");
  Serial.println("Type 't' for timer or 's' for stopwatch.");
}
```

3. Input the following loop() function:

```
void loop() {
  while(Serial.available() > 0)
  {
    char type = Serial.read(); // reads input from serial
    h = 0; m = 0; s = 0;
    // A newline character indicates end of input
    if(Serial.read() == '\n')
    {
      if (type == 's') // if input is s for stopwatch
      {
        Serial.println("You chose stopwatch.");
        startMillis = millis();
        while(digitalRead(Button2) == LOW) // loops while button 2 is not pushed
        {
          currentMillis = millis();
          elapsedMillis = (currentMillis - startMillis);
          // calculates h,m,s then print it
          h = (elapsedMillis / 3600000);
          m = ((elapsedMillis - (h*360000)) / 60000);
          s = ((elapsedMillis - ((m*60000)+(h*3600000))) / 1000);
          Serial.print(h);
          Serial.print(" hr ");
          Serial.print(m);
          Serial.print(" min ");
          Serial.print(s);
          Serial.print(" sec ");
          Serial.print(elapsedMillis - ((s*1000)+(m*60000)+(h*3600000)));
          Serial.println(" ms");
          if (digitalRead(Button1) == HIGH) // resets stopwatch from zero
          {
            startMillis = millis();
          }
        }
        Serial.println();
        Serial.println();
        Serial.println("Please decide whether you want to use the timer or the
stopwatch.");
        Serial.println("Type 't' for timer or 's' for stopwatch.");
      }
    }
  }
}
```

In this section, we read if timer or stopwatch is chosen in the Serial. Then, if stopwatch is chosen the code runs and prints the stopwatch times until the button at pin 4 is pressed to restart the stopwatch at zero. Once the button at pin 5 is pressed, we restart whether the user wants to use the timer or stopwatch.

4. Continue inputting the loop() function:

```

else if (type == 't') // if input is t for timer
{
  Serial.println("You chose timer.");
  Serial.println();
  Serial.println();
  Serial.println("Please enter the number of hours, minutes, and seconds
for the timer.");
  Serial.println("hours,minutes,seconds");
  Serial.println("ex: 0,5,5");
  Serial.println("This would start a timer for 5 minutes and 5 seconds");
  while(digitalRead(Button2) == LOW) // loops while button 2 is not pushed
  {
    while(Serial.available() > 0)
    {
      // reads input from serial into integers
      int hour = Serial.parseInt();
      int minute = Serial.parseInt();
      int second = Serial.parseInt();
      // A newline character indicates end of input
      if(Serial.read() == '\n')
      {
        // Adjust the values if out of range
        hour = constrain(hour, 0, 23);
        minute = constrain(minute, 0, 59);
        second = constrain(second, 0, 59);
        Serial.print("Setting the timer with");
        Serial.print(" hours=");
        Serial.print(hour, DEC);
        Serial.print(" minutes=");
        Serial.print(minute, DEC);
        Serial.print(" seconds=");
        Serial.println(second, DEC);
      }
    }
  }
}

```

If timer is chosen, then we ask the user for additional input for how long to set the timer. Then, we set the timer with chosen amount of time.

5. Input the final part of the loop() function:

```
int totalTime = (hour*3600000) + (minute*60000) + (second*1000);
int curTime = millis();
int finalTime = totalTime + curTime;
int remainingTime = 0;
while(finalTime >= curTime) // counts backwards by subtracting the
curTime from totalTime
{
    curTime = millis();
    remainingTime = finalTime - curTime;
    h = (remainingTime / 3600000);
    m = ((remainingTime - (h*3600000)) / 60000);
    s = ((remainingTime - ((m*60000)+(h*3600000))) / 1000);
    Serial.print(h);
    Serial.print(" hr ");
    Serial.print(m);
    Serial.print(" min ");
    Serial.print(s);
    Serial.println(" sec");
}
myTone(BUZZER, 500, 2000); // plays buzzer sound when timer is complete
Serial.println();
Serial.println();
Serial.println("Please enter the number of hours, minutes, and seconds
for the timer.");
Serial.println("hours,minutes,seconds");
Serial.println("ex: 0,5,5");
Serial.println("This would start a timer for 5 minutes and 5 seconds");
}
}
}
Serial.println();
Serial.println();
Serial.println("Please decide whether your want to use the timer or the
stopwatch.");
Serial.println("Type 't' for timer or 's' for stopwatch.");
}
else // if input is not t or s
{
    Serial.println("Not a valid input. Type either 't' or 's' to continue.");
}
}
}
```

This is where the timer prints out the remaining time and plays the sound when the timer is over. When the buzzer sounds, the user is given the choice of timer or stopwatch again. Also, if the user chooses other than timer 't' or stopwatch 's' we tell them to input their selection again.

6. Add the myTone function below and outside of the loop function to play a sound from the buzzer:

```
// plays the tone we want by using the vibrations of the buzzer
void myTone(byte pin, uint16_t frequency, uint16_t duration)
{ // input parameters: Arduino pin number, frequency in Hz, duration in milliseconds
  unsigned long startTime=millis();
  unsigned long halfPeriod= 1000000L/frequency/2;
  while (millis()-startTime< duration)
  {
    digitalWrite(pin,HIGH);
    delayMicroseconds(halfPeriod);
    digitalWrite(pin,LOW);
    delayMicroseconds(halfPeriod);
  }
}
```

Result

Open the Serial Monitor to choose whether you want to use the timer or stopwatch and input the according values to see your program at work. Please be sure to set the baud rate to 9600!

Now It's Your Turn!

- Can you make the stopwatch reset after a certain amount of time has elapsed?
- Can you make a watchdog timer where if you do not press a button within a certain amount of timer the buzzer sounds and the Neopixel flashes red?

Lesson I – 11: Bluetooth LED Control



OVERVIEW

In this lesson we will:

- Learn about Bluetooth and its applications
- Change LEDs on and off from connected Bluetooth mobile device
- Learn about Hexadecimal number representation system

Teacher Guide

For this lesson, students will be working with a mobile device and their Lenovo Educational Board to turn a LED on and off on the board using Bluetooth. On the mobile device, please download the LightBlue app (either Android or Apple Mobile device will work). After this is complete be sure to download the required libraries and we should be set up for this lesson.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

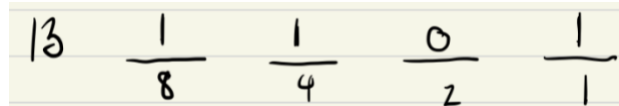
Background Info

What is Bluetooth? Bluetooth is used in most modern-day technologies to exchange data between multiple Bluetooth compatible devices. This technology uses short-range wireless radio transmissions over short distances to communicate with most Bluetooth connections working up to 30 feet between devices. In addition, for Bluetooth to work correctly devices must first be paired and connected to each other for security purposes before a packet is sent between the devices. A packet is information that is shipped between devices.

In this lesson, we will send Hexadecimal values between devices as out packet. Hexadecimal is a representation scheme to represent numbers which we use for things like counting and math and binary which is understood by computers. Every number can be represented in binary or hexadecimal by converting it. How to convert can be seen in the table. First, we must learn how to convert to binary which is then associated with a hexadecimal value. To represent a number in binary we must convert a number into 1s and 0s. For a number between 0 and 15 we can represent this with 4 bits which each hold a 1 or a 0. A 1 represents that value is present and 0 represents that that value is not present. Here is a 4-bit example for 13:

Decimal to Hexadecimal Table MATH

Decimal (Base 10)	Hexadecimal (Base 16)	Binary (Base 2)
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111



Procedure

1. Include the Bluetooth library:

```
#include <ArduinoBLE.h>
```

2. Above the setup() function create the following objects and variable:

```
BLEService ledService("19B10000-E8F2-537E-4F6C-D104768A1214"); // Bluetooth LED Information

BLEByteCharacteristic switchCharacteristic("19B10001-E8F2-537E-4F6C-D104768A1214", BLERead | BLEWrite); // Bluetooth Readable and Writable UUID

const int led = 15; // Green LED pin
```

3. Begin inputting the following `setup()` function:

```
void setup() {
  // Begin serial at 115200 baud rate
  Serial.begin(115200);

  // Setup LED pin
  pinMode(led, OUTPUT);

  // Initialize Bluetooth
  BLE.begin();

  // Set discoverable name and UUID
  BLE.setLocalName("Lenovo Educational Board");
}
```

4. Finish inputting the following `setup()` function:

```
  // Setup Bluetooth connections correctly and write value of 0
  BLE.setAdvertisedService(ledService);
  ledService.addCharacteristic(switchCharacteristic);
  BLE.addService(ledService);
  switchCharacteristic.writeValue(0);
  BLE.advertise();
  Serial.println("Bluetooth correctly initialized.");
}
```

Here we setup the Bluetooth for the Lenovo Educational Board and allow other devices to connect to it.

5. Begin inputting the following `loop()` function:

```
void loop() {
  // Listen for connecting Bluetooth devices
  BLEDevice central = BLE.central();

  // If devices are connected
  if (central) {
    // Print connected device MAC address
    Serial.print("Connected to device: ");
    Serial.println(central.address());
  }
}
```

We set up the Bluetooth device to connect to another device and display the MAC address when connected.

6. Continue inputting the `loop()` function:

```
// While device is connected
while (central.connected()) {
  // If value is written to Bluetooth
  if (switchCharacteristic.written()) {
    if (switchCharacteristic.value() == 1) { // A value of 1
      // Turn on LED
      Serial.println("LED on");
      digitalWrite(led, HIGH);
    } else if (switchCharacteristic.value() == 0) { // A 0 value
      // Turn off LED
      Serial.println("LED off");
      digitalWrite(led, LOW);
    }
  }
}
```

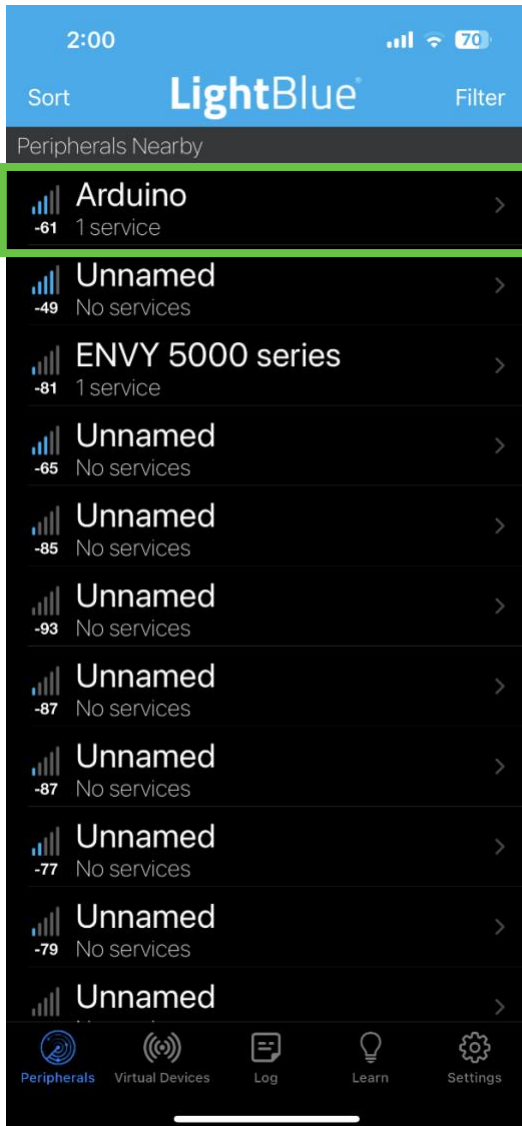
This is where we execute code if the device is connected to Bluetooth. While the device is connected to another, we determine what hexadecimal value is written to the device that is connected to the Lenovo Educational Board to control the LED.

7. Finish inputting the `loop()` function:

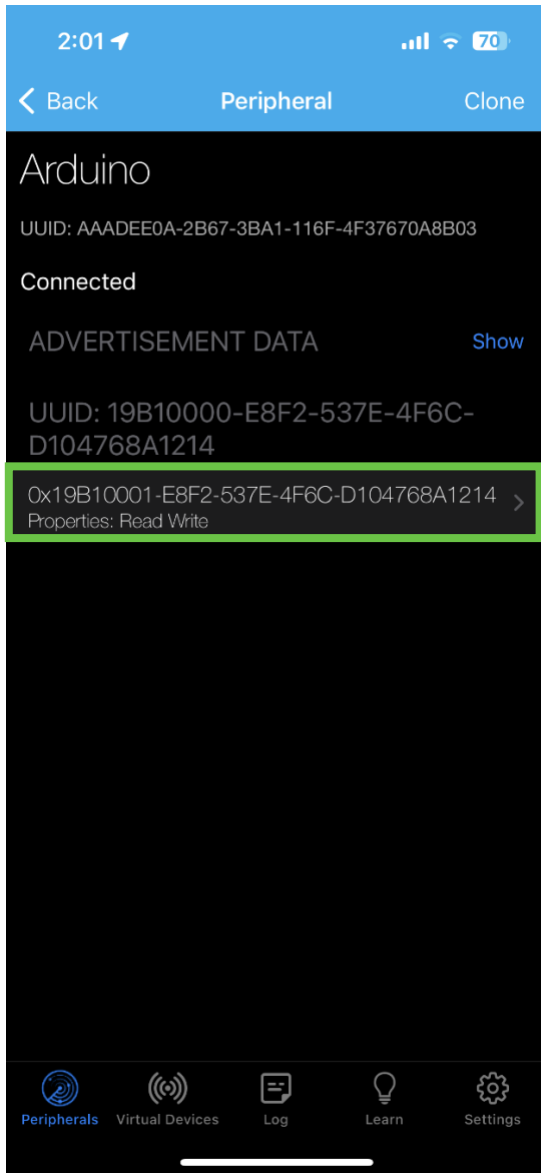
```
// If device disconnects print MAC address
Serial.print("Disconnected from device: ");
Serial.println(central.address());
}
```

8. Download the LightBlue app on your mobile device (either Android or iOS)

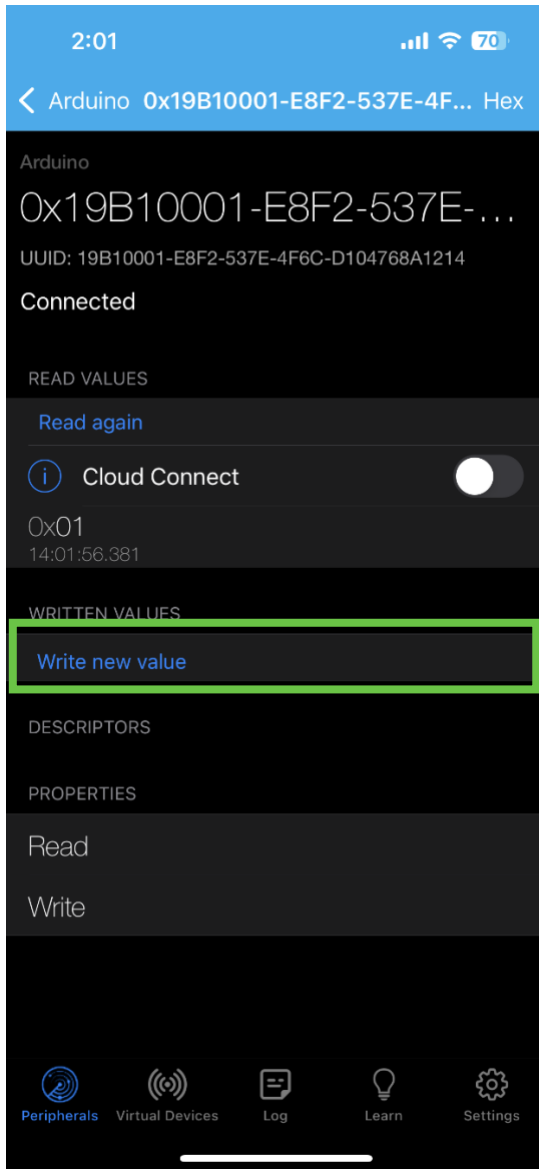
9. Open the app after uploading the sketch to the Lenovo Education Board and under Peripherals click "Arduino":



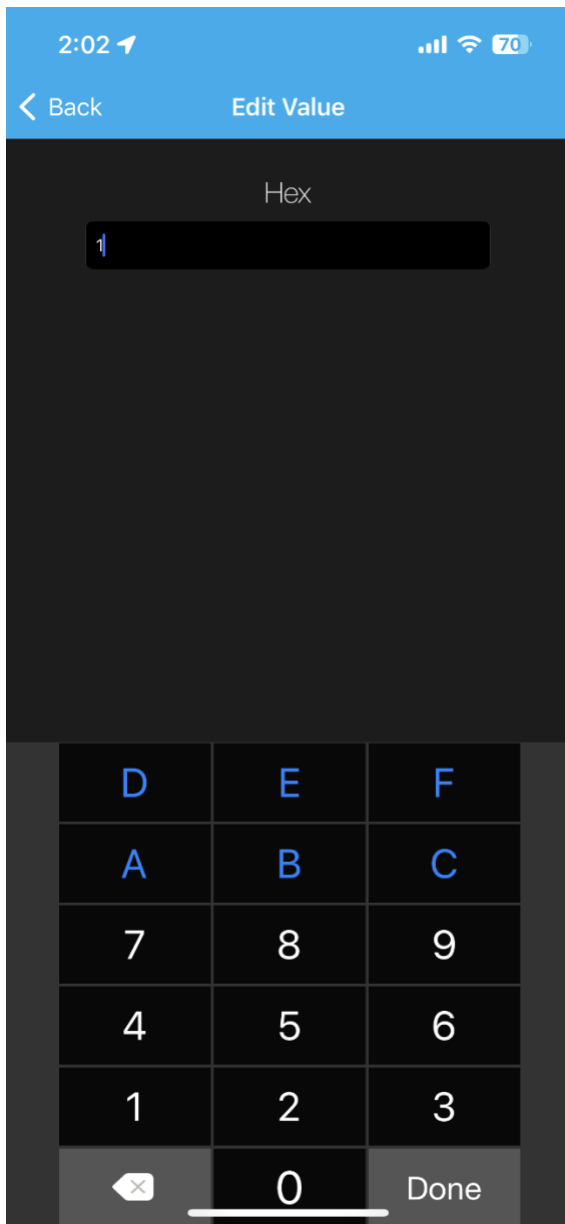
10. Click the Read/Write Button under Arduino:



11. Click "Write new Value":



12. Write a value of "1" to the app to turn the Green LED on and then repeat Step 11 to turn it off and write a value of 0:



Result

Now we have a LED that we can turn on and off over Bluetooth using our mobile device as a remote control.

Now It's Your Turn!

- Can you turn on/off more LEDs using other Hexadecimal values?
- Can you use Bluetooth to use different sensors to be displayed?

Lesson I – 12: OLED Display Temperature



OVERVIEW

In this lesson we will:

- Learn how to display information on the OLED display

Teacher Guide

In this lesson, students will require an OLED screen which can be purchased online at the following link for about \$3-5 per OLED screen: https://www.amazon.com/Hosyond-Display-Self-Luminous-Compatible-Raspberry/dp/B09C5K91H7/ref=pd_lpo_1?pd_rd_i=B09C5K91H7&psc=1

When attaching the OLED screen to the device please be sure to match the following pins to choose the correct header to input the OLED into. GND > GND / 3.3 V > VCC / SCL > SCL / SDA > SDA

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

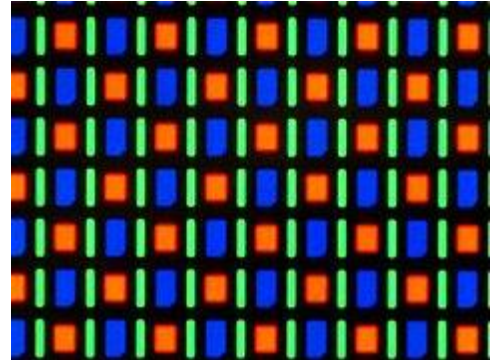
3. PIN number

Did you define the PIN number correctly in your program?

Background Info

What is an OLED display?

OLED stands for organic light emitting diode which is essentially a LED that uses a film of organic compound to emit light. For use in these lessons, when we refer to the OLED we are talking about the display. In recent years OLED screens have become increasingly popular as these use many tiny LEDs to create a picture with crisp detail. OLEDs are common on many everyday devices including tv screens, phone screens, and computer monitors.



Procedure

1. Include the following libraries, objects, and variable:

```
#include <Wire.h> // I2C library
#include "ens210.h" // ENS210 library
#include <SparkFun_APDS9960.h> // ambient light library
#include <SSD1306Wire.h> // OLED display library

// declare/initialize objects and variables
SSD1306Wire display(0x3c);
SparkFun_APDS9960 apds = SparkFun_APDS9960();
ENS210 ens210;
#define WIRE Wire
uint16_t ambient_light = 0;
```

2. Input the following `setup()` function:

```
void setup() {
  // Enable serial
  Serial.begin(9600);
  // initialize objects
  Wire.begin();
  ens210.begin();
  apds.init();
  apds.enableLightSensor(false);
  display.init();
  display.setContrast(255);
}
```

In the `setup()` function, we initialize the temperature sensor and the OLED display.

3. Input the following `loop()` function:

```
void loop() {
  // creates variables for temperature, humidity, and
  ambient light and measures new values
  int t_data, t_status, h_data, h_status;
  ens210.measure(&t_data, &t_status, &h_data, &h_status);
  apds.readAmbientLight(ambient_light);
  // prints output to OLED display
  display.setLogBuffer(5, 30);
  display.clear();
  display.print("T: ");
  display.print(ens210.toFahrenheit(t_data, 10) / 10.0);
  display.print("F , H: ");
  display.print(ens210.toPercentageH(h_data,1));
  display.println("%");
  display.print("Ambient Light: ");
  display.println(ambient_light);
  display.drawLogBuffer(0, 0);
  display.display();
  delay(2);
}
```

In this section, we receive the most recent data and print in on the OLED screen.

Result

Now, you can see the temperature, humidity, and ambient light data being displayed and updated on the OLED display.

Now It's Your Turn!

- Can you repeat previous lessons that we have completed and print to the OLED display instead?